

[illegible]

3

Sy

LI

LI
LILI
LILI
LILI
LILI
LI

LI

LI

LI

LI
LILI
LI

11

11

LI
LI

LI

LI

21

LI
LI

11

LI

LI

LI
LI

11

1

1

```

LL      IIIIII  BBBB BBBB  LL      EEEEEEEEE  XX      XX      IIIIII  CCCCCCCC  AAAAAA
LL      IIIIII  BBBB BBBB  LL      EEEEEEEEE  XX      XX      IIIIII  CCCCCCCC  AAAAAA
LL      II      BB      BB  LL      EE          XX      XX      II      CC      AA      AA
LL      II      BB      BB  LL      EE          XX      XX      II      CC      AA      AA
LL      II      BB      BB  LL      EE          XX      XX      II      CC      AA      AA
LL      II      BBBB BBBB  LL      EEEEEEEEE  XX      XX      II      CC      AA      AA
LL      II      BBBB BBBB  LL      EEEEEEEEE  XX      XX      II      CC      AA      AA
LL      II      BB      BB  LL      EE          XX      XX      II      CC      AA      AA
LL      II      BB      BB  LL      EE          XX      XX      II      CC      AA      AA
LL      II      BB      BB  LL      EE          XX      XX      II      CC      AA      AA
LLLLLLLL  IIIIII  BBBB BBBB  LLLLLLLLL  EEEEEEEEE  XX      XX      IIIIII  CCCCCCCC  AAAAAA
LLLLLLLL  IIIIII  BBBB BBBB  LLLLLLLLL  EEEEEEEEE  XX      XX      IIIIII  CCCCCCCC  AAAAAA

```



```

LL      IIIIII  SSSSSSSS
LL      IIIIII  SSSSSSSS
LL      II      SS
LL      II      SS
LL      II      SS
LL      II      SS
LL      II      SSSSSS
LL      II      SSSSSS
LL      II      SS
LL      II      SS
LL      II      SS
LLLLLLLL  IIIIII  SSSSSSSS
LLLLLLLL  IIIIII  SSSSSSSS

```



```
1 0001 0 MODULE LIB$$LEXICAL ( %TITLE 'Internal routines for lexical functions'
2 0002 0 IDENT = '1-009' ! File: LIBLEXICA.B32 Edit: STAN1009
3 0003 0 ) =
4 0004 1 BEGIN
5 0005 1
6 0006 1 *****
7 0007 1 *
8 0008 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
9 0009 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
10 0010 1 * ALL RIGHTS RESERVED.
11 0011 1 *
12 0012 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
13 0013 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
14 0014 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
15 0015 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
16 0016 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
17 0017 1 * TRANSFERRED.
18 0018 1 *
19 0019 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
20 0020 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
21 0021 1 * CORPORATION.
22 0022 1 *
23 0023 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
24 0024 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
25 0025 1 *
26 0026 1 *
27 0027 1 *****
28 0028 1
29 0029 1
30 0030 1 ++
31 0031 1 FACILITY: General Utility Library, DCL
32 0032 1
33 0033 1 ABSTRACT:
34 0034 1
35 0035 1 This module contains routines which form the common kernel of
36 0036 1 the following Run-Time Library procedures and DCL lexical functions:
37 0037 1 LIB$GETDVI F$GETDVI
38 0038 1 LIB$GETJPI F$GETJPI
39 0039 1 LIB$GETSYI F$GETSYI
40 0040 1
41 0041 1
42 0042 1 ENVIRONMENT: User or supervisor mode - AST reentrant
43 0043 1
44 0044 1 AUTHOR: Steven B. Lionel, CREATION DATE: 13-July-1982
45 0045 1
46 0046 1 MODIFIED BY:
47 0047 1
48 0048 1 1-001 - Original. Adapted from the DCL module LEXICON. SBL 13-July-1982
49 0049 1 1-002 - Use tables in LIBGETTAB.MAR. SBL 8-Mar-1983
50 0050 1 1-003 - Change string length from LNM$C NAMLENGTH to 512. SBL 11-Mar-1983
51 0051 1 1-004 - Add HEXSTR format. SBL 20-May-1983
52 0052 1 1-005 - HEXSTR is now HEXSTRING. SBL 24-May-1983
53 0053 1 1-006 - Add privileges TMPJNL, PRMJNL and SECURITY. SBL 28-July-1983
54 0054 1 1-007 - Add new format MODE for JPI$MODE. Fix format HEXSTRING so that
55 0055 1 the significant characters get returned. SBL 9-Sep-1983
56 0056 1 1-008 - Add support for two new arguments to SYS$GETSYI - NODENAME and
57 0057 1 CSIDADR. DG 19-Oct-1983.
```

LIB\$\$LEXICAL
1-009

Internal routines for lexical functions

1 3
16-Sep-1984 01:04:32
14-Sep-1984 12:39:06

VAX-11 Bliss-32 V4.0-742
[LIBRTL.SRC]LIBLEXICA.B32;1

Page 2
(1)

: 58
: 59
: 60

0058 1 ! 1-009 - Fix handling of counted strings. STAN 27-Feb-1984.
0059 1 !--
0060 1


```

: 62      0061 1 %SBTTL 'Declarations'
: 63      0062 1
: 64      0063 1 : PROLOGUE FILE:
: 65      0064 1 :
: 66      0065 1
: 67      0066 1 LIBRARY 'RTLILIB';           ! SYSS$LIBRARY:LIB.L32
: 68      0067 1 REQUIRE 'RTLIN:LIBPROLOG';    ! LIB$ definitions
: 69      0138 1
: 70      0139 1
: 71      0140 1 : LINKAGES:
: 72      0141 1 :
: 73      0142 1
: 74      0143 1 LINKAGE
: 75      0144 1     CALL_LEXICAL = CALL;
: 76      0145 1
: 77      0146 1 :
: 78      0147 1 : TABLE OF CONTENTS:
: 79      0148 1 :
: 80      0149 1
: 81      0150 1 FORWARD ROUTINE
: 82      0151 1     LIB$$GETDVI: CALL_LEXICAL, ! Get Device Information
: 83      0152 1     LIB$$GETJPI: CALL_LEXICAL, ! Get Job/Process Information
: 84      0153 1     LIB$$GETSYI: CALL_LEXICAL, ! Get System Information
: 85      0154 1     LIB$$FORMAT_RESULT: NOVALUE; ! Format result
: 86      0155 1
: 87      0156 1 :
: 88      0157 1 : MACROS:
: 89      0158 1 :
: 90      0159 1 :     NONE
: 91      0160 1 :
: 92      0161 1 : EQUATED SYMBOLS:
: 93      0162 1 :
: 94      0163 1 :     NONE
: 95      0164 1 :
: 96      0165 1 : FIELDS:
: 97      0166 1 :
: 98      0167 1 :     NONE
: 99      0168 1 :
: 100     0169 1 : OWN STORAGE:
: 101     0170 1 :
: 102     0171 1 :     NONE
: 103     0172 1 :
: 104     0173 1 : EXTERNAL REFERENCES:
: 105     0174 1 :
: 106     0175 1 :
: 107     0176 1 EXTERNAL ROUTINE
: 108     0177 1     OTSS$CVT_L_TZ;           ! Convert to hex format
: 109     0178 1
: 110     0179 1 EXTERNAL
: 111     0180 1     LIB$$AB_GETDVI_TABLE,      ! Table of $GETDVI codes and types
: 112     0181 1     LIB$$AB_GETJPI_TABLE,      ! Table of $GETJPI codes and types
: 113     0182 1     LIB$$AB_GETSYI_TABLE;      ! Table of $GETSYI codes and types
```

```

115 0183 1 XSBTTL 'LIB$$GETDVI - Internal routine for LIB$GETDVI'
116 0184 1 GLOBAL ROUTINE LIB$$GETDVI (
117 0185 1     ITEM_CODE: WORD SIGNED,      | $GETSYI Item code
118 0186 1     RET_STRING: REF VECTOR [, BYTE], | Return string buffer
119 0187 1     RET_NUMBER: REF BLOCK [, BYTE], | Return numeric buffer
120 0188 1     RET_LENGTH: REF VECTOR [, WORD], | Returned length
121 0189 1     RET_TYPE: REF VECTOR [, LONG], | Returned type code
122 0190 1     EVENT_FLAG,                | Event flag to use
123 0191 1     CHANNEL: WORD,              | Channel number
124 0192 1     DEVNAM_DESCR: REF BLOCK [, BYTE] | Device name descriptor
125 0193 1 ): CALL_LEXICAL =
126 0194 1
127 0195 1 ++
128 0196 1 FUNCTIONAL DESCRIPTION:
129 0197 1
130 0198 1     Kernel routine called from LIB$GETDVI and DCL to get device and
131 0199 1     volume information. See LIB$GETDVI for more information.
132 0200 1
133 0201 1 CALLING SEQUENCE:
134 0202 1
135 0203 1     ret-status.wlc.v = LIB$$GETDVI (
136 0204 1         item-code.rw.v,
137 0205 1         ret-string.wt.r,
138 0206 1         ret-number.wq.r,
139 0207 1         ret-length.wwu.r,
140 0208 1         ret-type.wl.r,
141 0209 1         event-flag.rl.v,
142 0210 1         channel.rwu.v,
143 0211 1         devnam-descr.rt.ds)
144 0212 1
145 0213 1 FORMAL PARAMETERS:
146 0214 1
147 0215 1     item-code          The $GETDVI item code
148 0216 1
149 0217 1     ret-string         A string of length 512 into which
150 0218 1                     is placed the string-formatted value.
151 0219 1
152 0220 1     ret-number         A quadword into which is placed the numeric
153 0221 1                     value, if any.
154 0222 1
155 0223 1     ret-length         A word into which is placed the length of
156 0224 1                     the string in ret-string.
157 0225 1
158 0226 1     ret-type           A longword into which is placed the type
159 0227 1                     code for the value being returned. The
160 0228 1                     codes are LIB$K_FMT_XXX values defined
161 0229 1                     in LIBFMTDEF.SDC.
162 0230 1
163 0231 1     event-flag         A longword event flag number to use for
164 0232 1                     the $GETDVI.
165 0233 1
166 0234 1     channel            A word containing the channel to inquire
167 0235 1                     about.
168 0236 1
169 0237 1     devnam-descr       A string descriptor for the device name
170 0238 1                     being inquired about.
171 0239 1
```


LIB\$\$LEXICAL
1-009

Internal routines for lexical functions
LIB\$\$GETDVI - Internal routine for LIB\$\$GETDVI

L 3
16-Sep-1984 01:04:32
14-Sep-1984 12:39:06

VAX-11 Bliss-32 V4.0-742
[LIBRTL.SRC]LIBLEXICA.B32;1

Page 5
(3)

```

: 172      0240 1 | IMPLICIT INPUTS:
: 173      0241 1 |
: 174      0242 1 |     NONE
: 175      0243 1 |
: 176      0244 1 | IMPLICIT OUTPUTS:
: 177      0245 1 |
: 178      0246 1 |     NONE
: 179      0247 1 |
: 180      0248 1 | COMPLETION STATUS:
: 181      0249 1 |
: 182      0250 1 |     $$$_NORMAL      Normal successful completion
: 183      0251 1 |     $$$_xxx         Any error status from $GETDVIW
: 184      0252 1 |
: 185      0253 1 | SIDE EFFECTS:
: 186      0254 1 |
: 187      0255 1 |     NONE
: 188      0256 1 |
: 189      0257 1 | --

```

LIB\$\$LEXICAL
1-009

Internal routines for lexical functions
LIB\$\$GETDVI - Internal routine for LIB\$GETDVI

M 3
16-Sep-1984 01:04:32
14-Sep-1984 12:39:06

VAX-11 Bliss-32 V4.0-742
[LIBRTL.SRC]LIBLEXICA.B32;1

Page 6
(4)

```
.. 191      0258 2      BEGIN
.. 192      0259 2
.. 193      0260 2
.. 194      0261 2      !+
.. 195      0262 2      ! Declare fieldset that defines the layout of a GETDVI_ITEM
.. 196      0263 2      !-
.. 197      0264 2      FIELD
.. 198      0265 2      GETDVI_ITEM_FIELDSET =
.. 199      0266 2      SET
.. 200      0267 2      W_ITEM = [0,0,16,1],      ! DVIS item code value
.. 201      0268 2      B_TYPE = [0,16,8,0],      ! LIB$R_FMT_type code
.. 202      0269 2      A_NEXT = [3,0,0,0]      ! Offset of next item
.. 203      0270 2      TES;
.. 204      0271 2
```



```
206 0272 2 LOCAL
207 0273 2 TABLE_ENTRY: REF BLOCK [, BYTE] FIELD (GETDVI_ITEM_FIELDSET),
208 0274 2 ! Current table entry
209 0275 2 DUMMY_ENTRY: BLOCK [3, BYTE] FIELD (GETDVI_ITEM_FIELDSET),
210 0276 2 ITEM_LIST: BLOCK [16, BYTE], ! Item list for $GETDVI
211 0277 2 IOSB: VECTOR [4, WORD], ! Status block
212 0278 2 RET_STATUS; ! Return status
213 0279 2
214 0280 2 !+
215 0281 2 ! Look up ITEM_CODE in LIB$$AB_GETDVI_TABLE.
216 0282 2 !-
217 0283 2
218 0284 2 TABLE_ENTRY = LIB$$AB_GETDVI_TABLE; ! Get first element.
219 0285 2
220 0286 2 WHILE .TABLE_ENTRY [W_ITEM] NEQ .ITEM_CODE
221 0287 2 DO
222 0288 2 BEGIN
223 0289 2 TABLE_ENTRY = TABLE_ENTRY [A_NEXT]; ! Get next item
224 0290 2 IF .TABLE_ENTRY [W_ITEM] EQL 0 ! No more items?
225 0291 2 THEN
226 0292 2 BEGIN
227 0293 2 TABLE_ENTRY = DUMMY_ENTRY; ! Use dummy table entry
228 0294 2 DUMMY_ENTRY [B_TYPE] = LIB$K_FMT_BINARY;
229 0295 2 EXITLOOP;
230 0296 2 END;
231 0297 2 END;
232 0298 2
233 0299 2 !+
234 0300 2 ! Store type code.
235 0301 2 !-
236 0302 2
237 0303 2 RET_TYPE [0] = .TABLE_ENTRY [B_TYPE];
238 0304 2
239 0305 2 !+
240 0306 2 ! Fill in ITEM_LIST and do the $GETDVI.
241 0307 2 !-
242 0308 2
243 0309 2 ITEM_LIST [0,16,16,0] = .ITEM_CODE; ! Item code
244 0310 2 IF .TABLE_ENTRY [B_TYPE] LEQ [LIB$K_FMT_MAXSTRING] ! Is it a string?
245 0311 2 THEN
246 0312 2 BEGIN
247 0313 2 ITEM_LIST [4,0,32,0] = RET_STRING [0]; ! Return buffer
248 0314 2 ITEM_LIST [0,0,16,0] = 512; ! Buffer size
249 0315 2 IF .TABLE_ENTRY [B_TYPE] EQL LIB$K_FMT_HEXSTRING
250 0316 2 THEN
251 0317 2 ITEM_LIST [0,0,16,0] = 256; ! Can't cvt more than 256 bytes
252 0318 2 END
253 0319 2 ELSE
254 0320 2 BEGIN
255 0321 2 RET_NUMBER [0,0,32,0] = 0; ! Zero the buffer
256 0322 2 RET_NUMBER [4,0,32,0] = 0;
257 0323 2 ITEM_LIST [4,0,32,0] = RET_NUMBER [0,0,0,0]; ! Return buffer
258 0324 2 ITEM_LIST [0,0,16,0] = 8; ! Buffer size (Quadword)
259 0325 2 END;
260 0326 2 ITEM_LIST [8,0,32,0] = RET_LENGTH [0]; ! Return length
261 0327 2 ITEM_LIST [12,0,32,0] = 0; ! End of list
262 0328 2
```

```
! End of routine LIB$$GETDVI
```

.ENTRY	LIB\$\$GETDVI, Save nothing	: 0184
SUBL2	#28, SP	: :
MOVAB	LIB\$\$AB GETDVI_TABLE, TABLE_ENTRY	: 0284
CMPW	(TABLE_ENTRY), ITEM_CODE	: 0286
BEQL	2\$: :
ADDL2	#3, TABLE_ENTRY	: 0289
TSTW	(TABLE_ENTRY)	: 0290
BNEQ	1\$: :
MOVAB	DUMMY_ENTRY, TABLE_ENTRY	: 0293
MOVB	#8, DUMMY_ENTRY+2	: 0294
MOVZBL	2(TABLE_ENTRY), @RET_TYPE	: 0303
MOVW	ITEM_CODE, ITEM_LIST+2	: 0309
CMPB	2(TABLE_ENTRY), #3	: 0310
BGTRU	3\$: :
MOVL	RET_STRING, ITEM_LIST+4	: 0313
MOVW	#512, ITEM_LIST	: 0314
CMPB	2(TABLE_ENTRY), #2	: 0315
BNEQ	4\$: :
MOVW	#256, ITEM_LIST	: 0317

LIB\$\$LEXICAL
1-009

Internal routines for lexical functions
LIB\$\$GETDVI - Internal routine for LIB\$GETDVI

C 4
16-Sep-1984 01:04:32
14-Sep-1984 12:39:06

VAX-11 Bliss-32 V4.0-742
[LIBRTL.SRC]LIBLEXICA.B32;1

Page 9
(5)

			0E	11	00047		BRB	4\$			
	50	0C	AC	D0	00049	3\$:	MOVL	RET_NUMBER, R0			0310
			60	7C	0004D		CLRQ	(R0)			0321
10	AE		50	D0	0004F		MOVL	R0, ITEM_LIST+4			0323
0C	AE		08	B0	00053		MOVW	#8, ITEM_LIST			0324
14	AE	10	AC	D0	00057	4\$:	MOVL	RET_LENGTH, ITEM_LIST+8			0326
		18	AE	D4	0005C		CLRL	ITEM_LIST+12			0327
			7E	7C	0005F		CLRQ	-(SP)			0330
			7E	D4	00061		CLRL	-(SP)			
		10	AE	9F	00063		PUSHAB	IOSB			
		1C	AE	9F	00066		PUSHAB	ITEM_LIST			
	7E	20	AC	DD	00069		PUSHL	DEVNAM_DESCR			
		1C	AC	3C	0006C		MOVZWL	CHANNEL, -(SP)			
00000000G	00	18	AC	DD	00070		PUSHL	EVENT_FLAG			
	17		08	FB	00073		CALLS	#8, SY\$\$GETDVIW			
	50	04	AE	3C	0007D		BLBC	RET_STATUS, 5\$			0332
	10		50	E9	00081		MOVZWL	IOSB, RET_STATUS			0334
	7E	10	AC	7D	00084		BLBC	RET_STATUS, 5\$			0340
	7E	08	AC	7D	00088		MOVQ	RET_LENGTH, -(SP)			0349
0000V	CF		04	FB	0008C		MOVQ	RET_STRING, -(SP)			
	50		01	D0	00091		CALLS	#4, LIB\$\$FORMAT_RESULT			
			04	D0	00094	5\$:	MOVL	#1, R0			0351
							RET				0353

; Routine Size: 149 bytes, Routine Base: _LIB\$CODE + 0000

```
289 0354 1 XSBTTL 'LIB$$GETJPI - Internal routine for LIB$$GETJPI'
290 0355 1 GLOBAL ROUTINE LIB$$GETJPI (
291 0356 1     ITEM CODE: WORD SIGNED,      | $GETJPI Item code
292 0357 1     RET_STRING: REF VECTOR [, BYTE], | Return string buffer
293 0358 1     RET_NUMBER: REF BLOCK [, BYTE], | Return numeric buffer
294 0359 1     RET_LENGTH: REF VECTOR [, WORD], | Returned length
295 0360 1     RET_TYPE: REF VECTOR [, LONG], | Returned type code
296 0361 1     EVENT_FLAG, | Event flag to use
297 0362 1     PIDADDR, | Address of PID
298 0363 1     PRCNAM_DESCR: REF BLOCK [, BYTE] | Process name descriptor
299 0364 1 ): CALL _LEXICAL =
300 0365 1
301 0366 1 ++
302 0367 1 FUNCTIONAL DESCRIPTION:
303 0368 1
304 0369 1     Kernel routine called from LIB$$GETJPI and DCL to get job and
305 0370 1     process information. See LIB$$GETJPI for more information.
306 0371 1
307 0372 1 CALLING SEQUENCE:
308 0373 1
309 0374 1     ret-status.wlc.v = LIB$$GETJPI (
310 0375 1         item-code.rw.v,
311 0376 1         ret-string.wt.r,
312 0377 1         ret-number.wq.r,
313 0378 1         ret-length.wwu.r,
314 0379 1         ret-type.wl.r,
315 0380 1         event-flag.rl.v,
316 0381 1         pidaddr.ra.v,
317 0382 1         prcnam-descr.rt.ds)
318 0383 1
319 0384 1 FORMAL PARAMETERS:
320 0385 1
321 0386 1     item-code      The $GETJPI item code
322 0387 1
323 0388 1     ret-string     A string of length 512 into which
324 0389 1                   is placed the string-formatted value.
325 0390 1
326 0391 1     ret-number     A quadword into which is placed the numeric
327 0392 1                   value, if any.
328 0393 1
329 0394 1     ret-length     A word into which is placed the length of
330 0395 1                   the string in ret-string.
331 0396 1
332 0397 1     ret-type       A longword into which is placed the type
333 0398 1                   code for the value being returned. The
334 0399 1                   codes are LIB$K_FMT_XXX values defined
335 0400 1                   in LIBFMTDEF.SDC.
336 0401 1
337 0402 1     event-flag     A longword event flag number to use for
338 0403 1                   the $GETJPI.
339 0404 1
340 0405 1     pidaddr        The address of the PID, if any, being inquired
341 0406 1                   about.
342 0407 1
343 0408 1     prcnam-descr   A string descriptor for the process name
344 0409 1                   being inquired about, if any
345 0410 1
```


LIB\$\$LEXICAL
1-009

Internal routines for lexical functions
LIB\$\$GETJPI - Internal routine for LIB\$\$GETJPI

E 4
16-Sep-1984 01:04:32
14-Sep-1984 12:39:06

VAX-11 Bliss-32 V4.0-742
[LIBRTL.SRC]LIBLEXICA.B32;1

Page 11
(6)

```
: 346      0411 1 | IMPLICIT INPUTS:
: 347      0412 1 |
: 348      0413 1 |     NONE
: 349      0414 1 |
: 350      0415 1 | IMPLICIT OUTPUTS:
: 351      0416 1 |
: 352      0417 1 |     NONE
: 353      0418 1 |
: 354      0419 1 | COMPLETION STATUS:
: 355      0420 1 |
: 356      0421 1 |     S$$_NORMAL      Normal successful completion
: 357      0422 1 |     S$$_xxx         Any error status from $GETJPIW
: 358      0423 1 |
: 359      0424 1 | SIDE EFFECTS:
: 360      0425 1 |
: 361      0426 1 |     NONE
: 362      0427 1 |
: 363      0428 1 | --
```

LIB\$\$LEXICAL
1-009

Internal routines for lexical functions
LIB\$\$GETJPI - Internal routine for LIB\$GETJPI

F 4
16-Sep-1984 01:04:32
14-Sep-1984 12:39:06

VAX-11 Bliss-32 V4.0-742
[LIBRTL.SRC]LIBLEXICA.B32;1

Page 12
(7)

```
: 365      0429  2  BEGIN
: 366      0430  2
: 367      0431  2  !+
: 368      0432  2  ! Declare fieldset that defines the layout of a GETJPI_ITEM.
: 369      0433  2  !-
: 370      0434  2
: 371      0435  2  FIELD
: 372      0436  2      GETJPI_ITEM_FIELDSET =
: 373      0437  2      SET
: 374      0438  2          W_ITEM = [0,0,16,1],      ! JPI$ item code value
: 375      0439  2          B_TYPE = [0,16,8,0],      ! LIB$R_FMT type code
: 376      0440  2          A_NEXT = [3,0,0,0]        ! Offset of next item
: 377      0441  2      TES;
: 378      0442  2
```



```

380      0443 2 LOCAL
381      0444 2 TABLE_ENTRY: REF BLOCK [, BYTE] FIELD (GETJPI_ITEM_FIELDSET),
382      0445 2 ! Current table entry
383      0446 2 DUMMY_ENTRY: BLOCK [3, BYTE] FIELD (GETJPI_ITEM_FIELDSET),
384      0447 2 ! Item list for $GETJPI
385      0448 2 IOSB: VECTOR [4, WORD], ! Status block
386      0449 2 RET_STATUS; ! Return status
387      0450 2
388      0451 2 !+
389      0452 2 ! Look up ITEM_CODE in LIB$$AB_GETJPI_TABLE.
390      0453 2 !-
391      0454 2
392      0455 2 TABLE_ENTRY = LIB$$AB_GETJPI_TABLE; ! Get first element.
393      0456 2
394      0457 2 WHILE .TABLE_ENTRY [W_ITEM] NEQ .ITEM_CODE
395      0458 2 DO
396      0459 2 BEGIN
397      0460 2 TABLE_ENTRY = TABLE_ENTRY [A_NEXT]; ! Get next item
398      0461 2 IF .TABLE_ENTRY [W_ITEM] EQL 0 ! No more items?
399      0462 2 THEN
400      0463 2 BEGIN
401      0464 2 TABLE_ENTRY = DUMMY_ENTRY; ! Use dummy entry
402      0465 2 TABLE_ENTRY [B_TYPE] = LIB$K_FMT_BINARY;
403      0466 2 EXITLOOP;
404      0467 2 END;
405      0468 2 END;
406      0469 2
407      0470 2 !+
408      0471 2 ! Store type code.
409      0472 2 !-
410      0473 2
411      0474 2 RET_TYPE [0] = .TABLE_ENTRY [B_TYPE];
412      0475 2
413      0476 2 !+
414      0477 2 ! Fill in ITEM_LIST and do the $GETJPI.
415      0478 2 !-
416      0479 2
417      0480 2 ITEM_LIST [0,16,16,0] = .ITEM_CODE; ! Item code
418      0481 2 IF .TABLE_ENTRY [B_TYPE] LEQ [LIB$K_FMT_MAXSTRING
419      0482 2 THEN
420      0483 2 BEGIN
421      0484 2 ITEM_LIST [4,0,32,0] = RET_STRING [0]; ! Return buffer
422      0485 2 ITEM_LIST [0,0,16,0] = 512; ! Buffer size
423      0486 2 IF .TABLE_ENTRY [B_TYPE] EQL LIB$K_FMT_HEXSTRING
424      0487 2 THEN
425      0488 2 ITEM_LIST [0,0,16,0] = 256; ! Can't cvt more than 256 bytes
426      0489 2 END
427      0490 2 ELSE
428      0491 2 BEGIN
429      0492 2 RET_NUMBER [0,0,32,0] = 0; ! Zero the buffer
430      0493 2 RET_NUMBER [4,0,32,0] = 0;
431      0494 2 ITEM_LIST [4,0,32,0] = RET_NUMBER [0,0,0,0]; ! Return buffer
432      0495 2 ITEM_LIST [0,0,16,0] = 8; ! Buffer size (Quadword)
433      0496 2 END;
434      0497 2 ITEM_LIST [8,0,32,0] = RET_LENGTH [0]; ! Return length
435      0498 2 ITEM_LIST [12,0,32,0] = 0; ! End of list
436      0499 2
```



```

: 437      P 0500 2      RET_STATUS = $GETJPIW (EFN = .EVENT_FLAG, PIDADR = .PIDADDR,
: 438      0501 2      -PRCNAM = PRCNAM_DESCR [0,0,0,0], ITMLST = ITEM_LIST, IOSB = IOSB);
: 439      0502 2
: 440      0503 2      IF .RET_STATUS
: 441      0504 2      THEN
: 442      0505 2      RET_STATUS = .IOSB [0];
: 443      0506 2
: 444      0507 2      !+
: 445      0508 2      ! Check for errors.
: 446      0509 2      !-
: 447      0510 2
: 448      0511 2      IF NOT .RET_STATUS
: 449      0512 2      THEN
: 450      0513 2      RETURN .RET_STATUS;
: 451      0514 2
: 452      0515 2      !+
: 453      0516 2      ! Now call LIB$$FORMAT_RESULT to format the result, if necessary.
: 454      0517 2      !-
: 455      0518 2
: 456      0519 2      LIB$$FORMAT_RESULT (RET_STRING [0], RET_NUMBER [0,0,0,0], RET_LENGTH [0],
: 457      0520 2      RET_TYPE [0]);
: 458      0521 2
: 459      0522 2      RETURN S$$_NORMAL;
: 460      0523 2
: 461      0524 1      END;

```

! End of routine LIB\$\$GETJPI

```

                                .EXTRN  SYSS$GETJPIW
                                0000 00000
                                1C  C2 00002
                                00  9E 00005
04  50 00000000G 60  B1 0000C 1$:
                                0E  13 00010
                                03  C0 00012
                                60  B5 00015
                                F3  12 00017
                                6E  9E 00019
02  A0          08  90 0001C
14  BC          02  A0 9A 00020 2$:
0E  AE          04  AC B0 00025
    03          02  A0 91 0002A
                                19  1A 0002E
                                AC  D0 00030
10  AE          08  AC D0 00030
0C  AE          0200 8F B0 00035
    02          02  A0 91 0003B
                                16  12 0003F
0C  AE          0100 8F B0 00041
                                0E  11 00047
    50          0C  AC D0 00049 3$:
                                60  7C 0004D
                                50  D0 0004F
10  AE          08  B0 00053
0C  AE          10  AC D0 00057 4$:
14  AE          18  AE D4 0005C
                                7E  7C 0005F
                                .ENTRY  LIB$$GETJPI, Save nothing
                                SUBL2  #28, SP
                                MOVAB  LIB$$AB_GETJPI_TABLE, TABLE_ENTRY
                                CMPW   (TABLE_ENTRY), -ITEM_CODE
                                BEQL   2$
                                ADDL2  #3, TABLE_ENTRY
                                TSTW   (TABLE_ENTRY)
                                BNEQ   1$
                                MOVAB  DUMMY_ENTRY, TABLE_ENTRY
                                MOVB   #8, 2(TABLE_ENTRY)
                                MOVZBL 2(TABLE_ENTRY), @RET_TYPE
                                MOVW   ITEM_CODE, ITEM_LIST+2
                                CMPB   2(TABLE_ENTRY), -#3
                                BGTRU  3$
                                MOVL   RET_STRING, ITEM_LIST+4
                                MOVW   #512, ITEM_LIST
                                CMPB   2(TABLE_ENTRY), #2
                                BNEQ   4$
                                MOVW   #256, ITEM_LIST
                                BRB    4$
                                MOVL   RET_NUMBER, R0
                                CLRQ   (R0)
                                MOVL   R0, ITEM_LIST+4
                                MOVW   #8, ITEM_LIST
                                MOVL   RET_LENGTH, ITEM_LIST+8
                                CLRL   ITEM_LIST+12
                                CLRQ   -(SP)

```


LIB\$\$LEXICAL
1-009

Internal routines for lexical functions
LIB\$\$GETJPI - Internal routine for LIB\$GETJPI

I 4
16-Sep-1984 01:04:32
14-Sep-1984 12:39:06

VAX-11 Bliss-32 V4.0-742
[LIBRTL.SRC]LIBLEXICA.B32;1

Page 15
(8)

		0C	AE	9F	00061	PUSHAB	IOSB	:	
		18	AE	9F	00064	PUSHAB	ITEM LIST	:	
	7E	1C	AC	7D	00067	MOVQ	PIDADDR, -(SP)	:	
		18	AC	DD	0006B	PUSHL	EVENT FLAG	:	
00000000G	00		07	FB	0006E	CALLS	#7, SYS\$GETJPIW	:	
	17		50	E9	00075	BLBC	RET STATUS, 5\$:	0503
	50	04	AE	3C	00078	MOVZWL	IOSB, RET STATUS	:	0505
	10		50	E9	0007C	BLBC	RET STATUS, 5\$:	0511
	7E	10	AC	7D	0007F	MOVQ	RET-LENGTH, -(SP)	:	0520
	7E	08	AC	7D	00083	MOVQ	RET-STRING, -(SP)	:	
0000V	CF		04	FB	00087	CALLS	#4, LIB\$\$FORMAT_RESULT	:	
	50		01	D0	0008C	MOVL	#1, R0	:	0522
			04	0008F	5\$: RET			:	0524

; Routine Size: 144 bytes, Routine Base: _LIB\$CODE + 0095

LIB\$\$LEXICAL
1-009

Internal routines for lexical functions
LIB\$\$GETSYI - Internal routine for LIB\$\$GETSYI

K 4
16-Sep-1984 01:04:32
14-Sep-1984 12:39:06

VAX-11 Bliss-32 V4.0-742
[LIBRTL.SRC]LIBLEXICA.B32;1

Page 17
(9)

```
.. 520      0582 1 | IMPLICIT INPUTS:
.. 521      0583 1 |
.. 522      0584 1 |     NONE
.. 523      0585 1 |
.. 524      0586 1 | IMPLICIT OUTPUTS:
.. 525      0587 1 |
.. 526      0588 1 |     NONE
.. 527      0589 1 |
.. 528      0590 1 | COMPLETION STATUS:
.. 529      0591 1 |
.. 530      0592 1 |     $$$_NORMAL      Normal successful completion
.. 531      0593 1 |     $$$_xxx         Any error status from $GETSYIW
.. 532      0594 1 |
.. 533      0595 1 | SIDE EFFECTS:
.. 534      0596 1 |
.. 535      0597 1 |     NONE
.. 536      0598 1 |
.. 537      0599 1 | --
```

LIB\$\$LEXICAL
1-009

Internal routines for lexical functions
LIB\$\$GETSYI - Internal routine for LIB\$GETSYI

L 4
16-Sep-1984 01:04:32
14-Sep-1984 12:39:06

VAX-11 Bliss-32 V4.0-742
[LIBRTL.SRC]LIBLEXICA.B32;1

Page 18
(10)

```

: 539      0600 2      BEGIN
: 540      0601 2
: 541      0602 2      !+
: 542      0603 2      !- Declare fieldset that defines the layout of a GETSYI_ITEM.
: 543      0604 2
: 544      0605 2
: 545      0606 2      FIELD
: 546      0607 2      GETSYI_ITEM_FIELDSET =
: 547      0608 2      SET
: 548      0609 2      W_ITEM = [0,0,16,0],      ! SYIS item code
: 549      0610 2      B_TYPE = [0,16,8,0],      ! LIB$R_FMT type code
: 550      0611 2      A_NEXT = [3,0,0,0]      ! Offset of next item
: 551      0612 2      TES;
: 552      0613 2
```



```
554 0614 2 LOCAL
555 0615 2 TABLE_ENTRY: REF BLOCK [, BYTE] FIELD (GETSYI_ITEM_FIELDSET),
556 0616 2 ! Current table entry
557 0617 2 DUMMY_ENTRY: BLOCK [3, BYTE] FIELD (GETSYI_ITEM_FIELDSET),
558 0618 2 ! I/O status block
559 0619 2 IOSB: VECTOR [4, WORD],
560 0620 2 ! Item list for $GETSYI
561 0621 2 RET_STATUS;
562 0622 2 ! Return status
563 0623 2 !+
564 0624 2 ! Look up ITEM_CODE in LIB$$AB_GETSYI_TABLE.
565 0625 2 !-
566 0626 2 TABLE_ENTRY = LIB$$AB_GETSYI_TABLE; ! Get first element.
567 0627 2
568 0628 2 WHILE .TABLE_ENTRY [W_ITEM] NEQ .ITEM_CODE
569 0629 2 DO
570 0630 2 BEGIN
571 0631 2 TABLE_ENTRY = TABLE_ENTRY [A_NEXT]; ! Get next item
572 0632 2 IF .TABLE_ENTRY [W_ITEM] EQL 0 ! No more items?
573 0633 2 THEN
574 0634 2 BEGIN
575 0635 2 TABLE_ENTRY = DUMMY_ENTRY; ! Use dummy entry
576 0636 2 TABLE_ENTRY [B_TYPE] = LIB$K_FMT_BINARY;
577 0637 2 EXITLOOP;
578 0638 2 END;
579 0639 2 END;
580 0640 2
581 0641 2 !+
582 0642 2 ! Store type code.
583 0643 2 !-
584 0644 2
585 0645 2 RET_TYPE [0] = .TABLE_ENTRY [B_TYPE];
586 0646 2
587 0647 2 !+
588 0648 2 ! Fill in ITEM_LIST and do the $GETSYI.
589 0649 2 !-
590 0650 2
591 0651 2 ITEM_LIST [0,16,16,0] = .ITEM_CODE; ! Item code
592 0652 2 IF .TABLE_ENTRY [B_TYPE] LEQ [LIB$K_FMT_MAXSTRING
593 0653 2 THEN
594 0654 2 BEGIN
595 0655 2 ITEM_LIST [4,0,32,0] = RET_STRING [0]; ! Return buffer
596 0656 2 ITEM_LIST [0,0,16,0] = 512; ! Buffer size
597 0657 2 IF .TABLE_ENTRY [B_TYPE] EQL LIB$K_FMT_HEXSTRING
598 0658 2 THEN
599 0659 2 ITEM_LIST [0,0,16,0] = 256; ! Can't cvt more than 256 bytes
600 0660 2 END
601 0661 2 ELSE
602 0662 2 BEGIN
603 0663 2 ITEM_LIST [4,0,32,0] = RET_NUMBER [0]; ! Return buffer
604 0664 2 ITEM_LIST [0,0,16,0] = 8; ! Buffer size (Quadword)
605 0665 2 END;
606 0666 2 ITEM_LIST [8,0,32,0] = RET_LENGTH [0]; ! Return length
607 0667 2 ITEM_LIST [12,0,32,0] = 0; ! End of list
608 0668 2
609 P 0669 2 RET_STATUS = $GETSYIW (EFN = .EVENT_FLAG, CSIDADR = .CSIDADR,
610 0670 2 NODENAME = .NODENAME_DESCR, ITMLST = ITEM_LIST, IOSB = IOSB);
```

```

: 611      0671 2
: 612      0672 2
: 613      0673 2
: 614      0674 2
: 615      0675 2
: 616      0676 2
: 617      0677 2
: 618      0678 2
: 619      0679 2
: 620      0680 2
: 621      0681 2
: 622      0682 2
: 623      0683 2
: 624      0684 2
: 625      0685 2
: 626      0686 2
: 627      0687 2
: 628      0688 2
: 629      0689 1

IF .RET_STATUS
THEN
    RET_STATUS = .IOSB [0];

IF NOT .RET_STATUS
THEN
    RETURN .RET_STATUS;

!+
!- Now call LIB$$FORMAT_RESULT to format the result.
!-

LIB$$FORMAT_RESULT (RET_STRING [0], RET_NUMBER [0], RET_LENGTH [0],
    RET_TYPE [0]);

RETURN SS$_NORMAL;

END;

! End of routine LIB$$GETSYI
```

```

                                .EXTRN  SYS$GETSYIW
                                .ENTRY   LIB$$GETSYI, Save nothing
                                SUBL2    #28, SP
                                MOVAB    LIB$$AB_GETSYI_TABLE, TABLE_ENTRY
                                CMPW     (TABLE_ENTRY), ITEM_CODE
                                BEQL     2$
                                ADDL2    #3, TABLE_ENTRY
                                TSTW     (TABLE_ENTRY)
                                BNEQ     1$
                                MOVAB    DUMMY_ENTRY, TABLE_ENTRY
                                MOVB     #8, 2(TABLE_ENTRY)
                                MOVZBL   2(TABLE_ENTRY), @RET_TYPE
                                MOVW     ITEM_CODE, ITEM_LIST+2
                                CMPB     2(TABLE_ENTRY), #3
                                BGTRU    3$
                                MOVL     RET_STRING, ITEM_LIST+4
                                MOVW     #512, ITEM_LIST
                                CMPB     2(TABLE_ENTRY), #2
                                BNEQ     4$
                                MOVW     #256, ITEM_LIST
                                BRB      4$
                                MOVL     RET_NUMBER, ITEM_LIST+4
                                MOVW     #8, ITEM_LIST
                                MOVL     RET_LENGTH, ITEM_LIST+8
                                CLRL     ITEM_LIST+12
                                CLRQ     -(SP)
                                PUSHAB   IOSB
                                PUSHAB   ITEM_LIST
                                MOVQ     CSIDADR, -(SP)
                                PUSHL    EVENT_FLAG
                                CALLS    #7, SYS$GETSYIW
                                BLBC     RET_STATUS, 5$
                                MOVZWL   IOSB, RET_STATUS
                                BLBC     RET_STATUS, 5$

                                0000 00000
                                1C  C2 00002
                                5E  50 00000000G 00 9E 00005
                                04  AC 60 B1 0000C 1$: 0E 13 00010
                                50 03 C0 00012
                                60 B5 00015
                                F3 12 00017
                                50 6E 9E 00019
                                02  A0 08 90 0001C
                                14  BC 02 A0 9A 00020 2$:
                                06  AE 04 AC B0 00025
                                03 02 A0 91 0002A
                                19 1A 0002E
                                08  AE 08 AC D0 00030
                                04  AE 0200 8F B0 00035
                                02 02 A0 91 0003B
                                11 12 0003F
                                04  AE 0100 8F B0 00041
                                09 11 00047
                                08  AE 0C AC D0 00049 3$:
                                04  AE 08 B0 0004E
                                0C  AE 10 AC D0 00052 4$:
                                10 AE D4 00057
                                7E 1C AE 9F 0005A
                                10 AE 9F 0005F
                                1C AC 7D 00062
                                18 AC DD 00066
                                00000000G 00 07 FB 00069
                                17 50 E9 00070
                                50 14 AE 3C 00073
                                10 50 E9 00077

                                0526
                                0626
                                0628
                                0631
                                0632
                                0635
                                0636
                                0645
                                0651
                                0652
                                0655
                                0656
                                0657
                                0659
                                0652
                                0663
                                0664
                                0666
                                0667
                                0670
                                0672
                                0674
                                0676
```


LIB\$\$LEXICAL
1-009

Internal routines for lexical functions
LIB\$\$GETSYI - Internal routine for LIB\$GETSYI

B 5
16-Sep-1984 01:04:32
14-Sep-1984 12:39:06

VAX-11 Bliss-32 V4.0-742
[LIBRTL.SRC]LIBLEXICA.B32;1

Page 21
(11)

	7E	10	AC	7D 0007A	MOVQ	RET_LENGTH, -(SP)	: 0685
	7E	08	AC	7D 0007E	MOVQ	RET_STRING, -(SP)	: :
0000V	CF		04	FB 00082	CALLS	#4, LIB\$\$FORMAT_RESULT	: :
	50		01	D0 00087	MOVL	#1, R0	: 0687
			04	0008A 5\$:	RET		: 0689

; Routine Size: 139 bytes, Routine Base: _LIB\$CODE + 0125


```
: 631 0690 1 %SBTTL 'LIB$$FORMAT_RESULT - Format the result'
: 632 0691 1 GLOBAL ROUTINE LIB$$FORMAT_RESULT (
: 633 0692 1     RET_STRING: REF VECTOR [, BYTE],      ! Return string buffer
: 634 0693 1     RET_NUMBER: REF BLOCK [, BYTE],      ! Return numeric buffer
: 635 0694 1     RET_LENGTH: REF VECTOR [, WORD],      ! Returned length
: 636 0695 1     RET_TYPE: REF VECTOR [, LONG]        ! Returned type code
: 637 0696 1 ): NOVALUE =
: 638 0697 1
: 639 0698 1 ++
: 640 0699 1 FUNCTIONAL DESCRIPTION:
: 641 0700 1
: 642 0701 1     Called by LIB$$GETxxI routines to convert the value returned
: 643 0702 1     by the $GETxxI service to the appropriate string format.
: 644 0703 1
: 645 0704 1 CALLING SEQUENCE:
: 646 0705 1
: 647 0706 1     CALL LIB$$FORMAT_RESULT (ret-string.mt.r, ret-number.rq.r,
: 648 0707 1     ret-length.mwu.r, ret-type.rl.r)
: 649 0708 1
: 650 0709 1 FORMAL PARAMETERS:
: 651 0710 1
: 652 0711 1     ret-string      A string of length 512 into which
: 653 0712 1                     is placed the formatted result.  If the
: 654 0713 1                     value type is already a string, the value
: 655 0714 1                     is in ret-string.
: 656 0715 1
: 657 0716 1     ret-number      A quadword containing the numeric value to
: 658 0717 1                     be formatted.
: 659 0718 1
: 660 0719 1     ret-length      A word containing the current length of the
: 661 0720 1                     string in ret-string, if any, and into which
: 662 0721 1                     is stored the length of the formatted result.
: 663 0722 1
: 664 0723 1     ret-type        A longword indicating the type of the value.
: 665 0724 1                     The type codes are LIB$K_FMT_xxx symbols and
: 666 0725 1                     are defined in LIBFMTDEF.SDL.
: 667 0726 1
: 668 0727 1 IMPLICIT INPUTS:
: 669 0728 1
: 670 0729 1     NONE
: 671 0730 1
: 672 0731 1 IMPLICIT OUTPUTS:
: 673 0732 1
: 674 0733 1     NONE
: 675 0734 1
: 676 0735 1 COMPLETION STATUS:
: 677 0736 1
: 678 0737 1     NONE
: 679 0738 1
: 680 0739 1 SIDE EFFECTS:
: 681 0740 1
: 682 0741 1     NONE
: 683 0742 1
: 684 0743 1 --
: 685 0744 1
: 686 0745 2 BEGIN
: 687 0746 2
```



```

688 0747 2 LOCAL
689 0748 2   CTRSTR_DESCR: BLOCK [8, BYTE], ! FAO control string descriptor
690 0749 2   OUTSTR_DESCR: BLOCK [8, BYTE], ! Output string descriptor
691 0750 2   PRMLST: VECTOR [4, LONG]; ! FAOL parameter list
692 0751 2
693 0752 2 !+
694 0753 2 ! Table of ACP type names.
695 0754 2 !-
696 0755 2
697 0756 2 BIND
698 0757 2   ACP_TYPES = UPLIT BYTE (
699 0758 2     %ASCIC'UNKNOWN', ! 0
700 0759 2     %ASCIC'F11V1', 0, 0, $ASSUME (DVISC_ACP_F11V1, EQL, 1)
701 0760 2     %ASCIC'F11V2', 0, 0, $ASSUME (DVISC_ACP_F11V2, EQL, 2)
702 0761 2     %ASCIC'MTA', 0, 0, 0, $ASSUME (DVISC_ACP_MTA, EQL, 3)
703 0762 2     %ASCIC'NET', 0, 0, 0, $ASSUME (DVISC_ACP_NET, EQL, 4)
704 0763 2     %ASCIC'REM', 0, 0, 0, $ASSUME (DVISC_ACP_REM, EQL, 5)
705 0764 2     %ASCIC'JNL', 0, 0, 0, $ASSUME (DVISC_ACP_JNL, EQL, 6)
706 0765 2     : VECTOR [, LONG];
707 0766 2
708 0767 2 !+
709 0768 2 ! Table of process state names.
710 0769 2 !-
711 0770 2
712 0771 2 BIND
713 0772 2   STATES = UPLIT BYTE (
714 0773 2     %ASCIC'UNKNOWN', ! 0
715 0774 2     %ASCIC'COLPG', 0, 0, $ASSUME (SCHSC_COLPG, EQL, 1)
716 0775 2     %ASCIC'MWAIT', 0, 0, $ASSUME (SCHSC_MWAIT, EQL, 2)
717 0776 2     %ASCIC'CEF', 0, 0, 0, $ASSUME (SCHSC_CEF, EQL, 3)
718 0777 2     %ASCIC'PFW', 0, 0, 0, $ASSUME (SCHSC_PFW, EQL, 4)
719 0778 2     %ASCIC'LEF', 0, 0, 0, $ASSUME (SCHSC_LEF, EQL, 5)
720 0779 2     %ASCIC'LEFO', 0, 0, 0, $ASSUME (SCHSC_LEFO, EQL, 6)
721 0780 2     %ASCIC'HIB', 0, 0, 0, $ASSUME (SCHSC_HIB, EQL, 7)
722 0781 2     %ASCIC'HIBO', 0, 0, 0, $ASSUME (SCHSC_HIBO, EQL, 8)
723 0782 2     %ASCIC'SUSP', 0, 0, 0, $ASSUME (SCHSC_SUSP, EQL, 9)
724 0783 2     %ASCIC'SUSPO', 0, 0, $ASSUME (SCHSC_SUSPO, EQL, 10)
725 0784 2     %ASCIC'FPG', 0, 0, 0, $ASSUME (SCHSC_FPG, EQL, 11)
726 0785 2     %ASCIC'COM', 0, 0, 0, $ASSUME (SCHSC_COM, EQL, 12)
727 0786 2     %ASCIC'COMO', 0, 0, 0, $ASSUME (SCHSC_COMO, EQL, 13)
728 0787 2     %ASCIC'CUR', 0, 0, 0, $ASSUME (SCHSC_CUR, EQL, 14)
729 0788 2     : VECTOR [, LONG];
730 0789 2
731 0790 2 !+
732 0791 2 ! Table of process mode names.
733 0792 2 !-
734 0793 2
735 0794 2 BIND
736 0795 2   MODES = UPLIT BYTE (
737 0796 2     %ASCIC'OTHER', $ASSUME (JPI$K_OTHER, EQL, 0)
738 0797 2     %ASCIC'NETWORK', $ASSUME (JPI$K_NETWORK, EQL, 1)
739 0798 2     %ASCIC'BATCH', $ASSUME (JPI$K_BATCH, EQL, 2)
740 0799 2     %ASCIC'INTERACTIVE', $ASSUME (JPI$K_INTERACTIVE, EQL, 3)
741 0800 2     0) ! End of list
742 0801 2     : VECTOR [, BYTE];
743 0802 2
744 0803 2 !+
```

```
: 745      0804 2      ! Table of privilege names.
: 746      0805 2      !-
: 747      0806 2
: 748      0807 2      GLOBAL BIND
: 749      0808 2      LIB$SAT PRV NAMES = UPLIT BYTE (
: 750      0809 2      %ASCIC'CMKRNL', $ASSUME ($BITPOSITION(PRV$V_CMKRNL), EQL, 0)
: 751      0810 2      %ASCIC'CMEXEC', $ASSUME ($BITPOSITION(PRV$V_CMEXEC), EQL, 1)
: 752      0811 2      %ASCIC'SYSNAM', $ASSUME ($BITPOSITION(PRV$V_SYSNAM), EQL, 2)
: 753      0812 2      %ASCIC'GRPNAM', $ASSUME ($BITPOSITION(PRV$V_GRPNAM), EQL, 3)
: 754      0813 2      %ASCIC'ALLSPOOL', $ASSUME ($BITPOSITION(PRV$V_ALLSPOOL), EQL, 4)
: 755      0814 2      %ASCIC'DETACH', $ASSUME ($BITPOSITION(PRV$V_DETACH), EQL, 5)
: 756      0815 2      %ASCIC'DIAGNOSE', $ASSUME ($BITPOSITION(PRV$V_DIAGNOSE), EQL, 6)
: 757      0816 2      %ASCIC'LOG IO', $ASSUME ($BITPOSITION(PRV$V_LOG IO), EQL, 7)
: 758      0817 2      %ASCIC'GROUP', $ASSUME ($BITPOSITION(PRV$V_GROUP), EQL, 8)
: 759      0818 2      %ASCIC'NOACNT', $ASSUME ($BITPOSITION(PRV$V_NOACNT), EQL, 9)
: 760      0819 2      %ASCIC'PRMCEB', $ASSUME ($BITPOSITION(PRV$V_PRMCEB), EQL, 10)
: 761      0820 2      %ASCIC'PRMMBX', $ASSUME ($BITPOSITION(PRV$V_PRMMBX), EQL, 11)
: 762      0821 2      %ASCIC'PSWAPM', $ASSUME ($BITPOSITION(PRV$V_PSWAPM), EQL, 12)
: 763      0822 2      %ASCIC'SETPRI', $ASSUME ($BITPOSITION(PRV$V_SETPRI), EQL, 13)
: 764      0823 2      %ASCIC'SETPRV', $ASSUME ($BITPOSITION(PRV$V_SETPRV), EQL, 14)
: 765      0824 2      %ASCIC'TMPMBX', $ASSUME ($BITPOSITION(PRV$V_TMPMBX), EQL, 15)
: 766      0825 2      %ASCIC'WORLD', $ASSUME ($BITPOSITION(PRV$V_WORLD), EQL, 16)
: 767      0826 2      %ASCIC'MOUNT', $ASSUME ($BITPOSITION(PRV$V_MOUNT), EQL, 17)
: 768      0827 2      %ASCIC'OPER', $ASSUME ($BITPOSITION(PRV$V_OPER), EQL, 18)
: 769      0828 2      %ASCIC'EXQUOTA', $ASSUME ($BITPOSITION(PRV$V_EXQUOTA), EQL, 19)
: 770      0829 2      %ASCIC'NETMBX', $ASSUME ($BITPOSITION(PRV$V_NETMBX), EQL, 20)
: 771      0830 2      %ASCIC'VOLPRO', $ASSUME ($BITPOSITION(PRV$V_VOLPRO), EQL, 21)
: 772      0831 2      %ASCIC'PHY IO', $ASSUME ($BITPOSITION(PRV$V_PHY IO), EQL, 22)
: 773      0832 2      %ASCIC'BUGCHK', $ASSUME ($BITPOSITION(PRV$V_BUGCHK), EQL, 23)
: 774      0833 2      %ASCIC'PRMGBL', $ASSUME ($BITPOSITION(PRV$V_PRMGBL), EQL, 24)
: 775      0834 2      %ASCIC'SYSGBL', $ASSUME ($BITPOSITION(PRV$V_SYSGBL), EQL, 25)
: 776      0835 2      %ASCIC'PFNMAP', $ASSUME ($BITPOSITION(PRV$V_PFNMAP), EQL, 26)
: 777      0836 2      %ASCIC'SHMEM', $ASSUME ($BITPOSITION(PRV$V_SHMEM), EQL, 27)
: 778      0837 2      %ASCIC'SYSPRV', $ASSUME ($BITPOSITION(PRV$V_SYSPRV), EQL, 28)
: 779      0838 2      %ASCIC'BYPASS', $ASSUME ($BITPOSITION(PRV$V_BYPASS), EQL, 29)
: 780      0839 2      %ASCIC'SYSLCK', $ASSUME ($BITPOSITION(PRV$V_SYSLCK), EQL, 30)
: 781      0840 2      %ASCIC'SHARE', $ASSUME ($BITPOSITION(PRV$V_SHARE), EQL, 31)
: 782      0841 2      %ASCIC'UPGRADE', $ASSUME ($BITPOSITION(PRV$V_UPGRADE), EQL, 0) ! 32
: 783      0842 2      %ASCIC'DOWNGRADE', $ASSUME ($BITPOSITION(PRV$V_DOWNGRADE), EQL, 1) ! 33
: 784      0843 2      %ASCIC'GRPPRV', $ASSUME ($BITPOSITION(PRV$V_GRPPRV), EQL, 2) ! 34
: 785      0844 2      %ASCIC'READALL', $ASSUME ($BITPOSITION(PRV$V_READALL), EQL, 3) ! 35
: 786      0845 2      %ASCIC'TMPJNL', $ASSUME ($BITPOSITION(PRV$V_TMPJNL), EQL, 4) ! 36
: 787      0846 2      %ASCIC'PRMJNL', $ASSUME ($BITPOSITION(PRV$V_PRMJNL), EQL, 5) ! 37
: 788      0847 2      %ASCIC'SECURITY', $ASSUME ($BITPOSITION(PRV$V_SECURITY), EQL, 6) ! 38
: 789      0848 2      0) ! End of list
: 790      0849 2      : VECTOR [, BYTE];
: 791      0850 2
: 792      0851 2      !+
: 793      0852 2      ! Fill in constant descriptor information.
: 794      0853 2      !-
: 795      0854 2
: 796      0855 2      CTRSTR_DESCR [DSC$B_DTYPE] = DSC$K_DTYPE_T;
: 797      0856 2      CTRSTR_DESCR [DSC$B_CLASS] = DSC$K_CLASS_S;
: 798      0857 2      OUTSTR_DESCR [DSC$B_DTYPE] = DSC$K_DTYPE_T;
: 799      0858 2      OUTSTR_DESCR [DSC$B_CLASS] = DSC$K_CLASS_S;
: 800      0859 2      OUTSTR_DESCR [DSC$W_LENGTH] = 512;
: 801      0860 2      OUTSTR_DESCR [DSC$A_POINTER] = REF_STRING [0];
```



```

802 0861
803 0862
804 0863
805 0864
806 0865
807 0866
808 0867
809 0868
810 0869
811 0870
812 0871
813 0872
814 0873
815 0874
816 0875
817 0876
818 0877
819 0878
820 0879
821 0880
822 0881
823 0882
824 0883
825 0884
826 0885
827 0886
828 0887
829 0888
830 0889
831 0890
832 0891
833 0892
834 0893
835 0894
836 0895
837 0896
838 0897
839 0898
840 0899
841 0900
842 0901
843 0902
844 0903
845 0904
846 0905
847 0906
848 0907
849 0908
850 0909
851 0910
852 0911
853 0912
854 0913
855 0914
856 0915
857 0916
858 0917

```

```

!+
!-
Select the formatting action appropriate to the item type.

CASE .RET_TYPE [0] FROM LIB$K_FMT_MIN TO LIB$K_FMT_MAX OF
SET

[LIB$K_FMT_BINARY]:
BEGIN
CTRSTR_DESCR [DSC$W_LENGTH] = %CHARCOUNT ('!UL');
CTRSTR_DESCR [DSC$A_POINTER] = UPLIT BYTE ('!UL');
PRMLST [0] = .RET_NUMBER [0,0,32,0];
$FAOL (CTRSTR = CTRSTR_DESCR [0,0,0,0],
OUTLEN = RET_LENGTH [0],
OUTBUF = OUTSTR_DESCR [0,0,0,0],
PRMLST = PRMLST [0]);
END;

[LIB$K_FMT_BOOLEAN]:
BEGIN
IF .RET_NUMBER [0,0,1,0]
THEN
BEGIN
RET_LENGTH [0] = %CHARCOUNT ('TRUE');
CH$MOVE (%CHARCOUNT ('TRUE'), UPLIT BYTE ('TRUE'),
RET_STRING [0]);
END
ELSE
BEGIN
RET_LENGTH [0] = %CHARCOUNT ('FALSE');
CH$MOVE (%CHARCOUNT ('FALSE'), UPLIT BYTE ('FALSE'),
RET_STRING [0]);
END;
END;

[LIB$K_FMT_HEX]:
BEGIN
CTRSTR_DESCR [DSC$W_LENGTH] = %CHARCOUNT ('!XL');
CTRSTR_DESCR [DSC$A_POINTER] = UPLIT BYTE ('!XL');
PRMLST [0] = .RET_NUMBER [0,0,32,0];
$FAOL (CTRSTR = CTRSTR_DESCR [0,0,0,0],
OUTLEN = RET_LENGTH [0],
OUTBUF = OUTSTR_DESCR [0,0,0,0],
PRMLST = PRMLST [0]);
END;

[LIB$K_FMT_HEXSTRING]:
BEGIN
LOCAL
TEMP_STRING: VECTOR [512, BYTE];

CTRSTR_DESCR [DSC$W_LENGTH] = .RET_LENGTH [0] * 2;
CTRSTR_DESCR [DSC$A_POINTER] = TEMP_STRING;
OT$SCVT_L_T2 (RET_STRING [0], CTRSTR_DESCR [0,0,0,0],
CTRSTR_DESCR [DSC$W_LENGTH], .RET_LENGTH [0]);
RET_LENGTH [0] = .RET_LENGTH [0] * 2;

```

```

859      0918 3      CH$MOVE (.CTRSTR_DESCR [DSC$W_LENGTH], TEMP_STRING,
860      0919      RET_STRING [0]);
861      0920      END;
862      0921
863      0922 [LIB$K_FMT_DATE]:
864      0923 BEGIN
865      0924 CTRSTR_DESCR [DSC$W_LENGTH] = %CHARCOUNT ('!%D');
866      0925 CTRSTR_DESCR [DSC$A-POINTER] = UPLIT BYTE ('!%D');
867      0926 PRMLST [0] = RET_NUMBER [0,0,0,0];
868      0927 $FAOL (CTRSTR = CTRSTR_DESCR [0,0,0,0],
869      0928 OUTLEN = RET_LENGTH [0],
870      0929 OUTBUF = OUTSTR_DESCR [0,0,0,0],
871      0930 PRMLST = PRMLST [0]);
872      0931 END;
873      0932
874      0933 [LIB$K_FMT_PRIVILEGE]:
875      0934 BEGIN
876      0935 LOCAL
877      0936 STRING_PTR, ! Pointer to current char in string
878      0937 PRV_NAME: REF VECTOR [, BYTE], ! Privilege name
879      0938 PRV; ! Current privilege number
880      0939
881      0940 STRING_PTR = RET_STRING [0]; ! First position in string
882      0941 PRV_NAME = LIB$$AT PRV_NAMES [0]; ! First privilege name
883      0942 INCRU PRV FROM 0 TO 63 DO
884      0943 BEGIN
885      0944 IF .PRV_NAME [0] EQL 0 ! No more defined privilege names
886      0945 THEN
887      0946 EXITLOOP;
888      0947 IF .RET_NUMBER [0,.PRV,1,0]
889      0948 THEN
890      0949 BEGIN
891      0950 STRING_PTR = CH$MOVE (.PRV_NAME [0], PRV_NAME [1],
892      0951 .STRING_PTR);
893      0952 CH$WCHAR_A (%C',, STRING_PTR);
894      0953 END;
895      0954 PRV_NAME = .PRV_NAME + .PRV_NAME [0] + 1; ! Next name
896      0955 END;
897      0956 IF .STRING_PTR NEQA RET_STRING [0]
898      0957 THEN
899      0958 STRING_PTR = .STRING_PTR - 1; ! Trim trailing comma
900      0959 RET_LENGTH [0] = .STRING_PTR - RET_STRING [0]; ! Get length
901      0960 END;
902      0961
903      0962 [LIB$K_FMT_UIC]:
904      0963 BEGIN
905      0964 CTRSTR_DESCR [DSC$W_LENGTH] = %CHARCOUNT ('!%U');
906      0965 CTRSTR_DESCR [DSC$A-POINTER] = UPLIT BYTE ('!%U');
907      0966 PRMLST [0] = .RET_NUMBER [0,0,32,0];
908      0967 $FAOL (CTRSTR = CTRSTR_DESCR [0,0,0,0],
909      0968 OUTLEN = RET_LENGTH [0],
910      0969 OUTBUF = OUTSTR_DESCR [0,0,0,0],
911      0970 PRMLST = PRMLST [0]);
912      0971 END;
913      0972
914      0973 [LIB$K_FMT_PROT, LIB$K_FMT_VPROT]:
915      0974 BEGIN
```



```

916      0975      3      LOCAL
917      0976      PSTRING: VECTOR [24, BYTE],
918      0977      PSTRING_PTR,
919      0978      PROT_CHARS: REF VECTOR [, BYTE],
920      0979      PROT_FIELD: BLOCK [1, BYTE];
921      0980
922      0981      !+
923      0982      !- Select the correct protection codes for files or volumes.
924      0983
925      0984
926      0985      IF .RET_TYPE EQL LIB$K_FMT_PROT
927      0986      THEN
928      0987          PROT_CHARS = UPLIT BYTE ('RWED')
929      0988
930      0989      ELSE
931      0990          PROT_CHARS = UPLIT BYTE ('RWLP');
932      0991
933      0992      PSTRING_PTR = PSTRING [0];
934      0993      INCR I FROM 0 TO 3 BY 1 DO
935      0994          BEGIN
936      0995              LOCAL
937      0996                  THIS_STRING: REF VECTOR [, BYTE];
938      0997                  PRMLST [I] = .PSTRING_PTR;
939      0998                  THIS_STRING = .PSTRING_PTR;
940      0999                  CH$WCHAR A (0, PSTRING_PTR);          ! Set initial length
941      1000                  PROT_FIELD = .RET_NUMBER [0, I*4, 4, 0] XOR 'X'F';
942      1001                  IF .PROT_FIELD NEQ 0
943      1002                      THEN
944      1003                          BEGIN
945      1004                              CH$WCHAR A (%C=' ', PSTRING_PTR);
946      1005                              THIS_STRING [0] = .THIS_STRING [0] + 1;
947      1006                              INCR J FROM 0 TO 3 BY 1 DO
948      1007                                  BEGIN
949      1008                                      IF .PROT_FIELD [0, J, 1, 0]
950      1009                                          THEN
951      1010                                              BEGIN
952      1011                                                  CH$WCHAR A (.PROT_CHARS [J], PSTRING_PTR);
953      1012                                                  THIS_STRING [0] = .THIS_STRING [0] + T;
954      1013                                                  END;
955      1014                                              END;
956      1015                                  END;
957      1016                              END;
958      1017                          END;
959      1018                  CTRSTR_DESCR [DSC$W_LENGTH] =
960      1019                      %CHARCOUNT ('SYSTEM!AC,OWNER!AC,GROUP!AC,WORLD!AC');
961      1020                  CTRSTR_DESCR [DSC$A_POINTER] =
962      1021                      UPLIT BYTE ('SYSTEM!AC,OWNER!AC,GROUP!AC,WORLD!AC');
963      1022                  $FAOL (CTRSTR = CTRSTR_DESCR [0,0,0,0],
964      1023                      OUTLEN = RET_LENGTH [0],
965      1024                      OUTBUF = OUTSTR_DESCR [0,0,0,0],
966      1025                      PRMLST = PRMLST [0]);
967      1026                  END;
968      1027      [LIB$K_FMT_ACP]:
969      1028      BEGIN
970      1029          LOCAL
971      1030              ACPTYP_PTR: REF VECTOR [, BYTE];
972      1031              IF .RET_NUMBER [0,0,32,0] GTRU DVI$C_ACP_JNL
```

```

: 973      1032      3      THEN
: 974      1033      3      ACPTYP_PTR = ACP_TYPES [0]      ! Illegal
: 975      1034      3      ELSE
: 976      1035      3      ACPTYP_PTR = ACP_TYPES [2*.RET_NUMBER [0,0,32,0]];
: 977      1036      3      RET_LENGTH [0] = .ACPTYP_PTR [0];
: 978      1037      3      CH$MOVE (.RET_LENGTH [0], ACPTYP_PTR [1], RET_STRING [0]);
: 979      1038      3      END;
: 980      1039      3
: 981      1040      3      [LIB$K_FMT_STATE]:
: 982      1041      3      BEGIN
: 983      1042      3      LOCAL
: 984      1043      3      STATE_PTR: REF VECTOR [, BYTE];
: 985      1044      3      IF .RET_NUMBER [0,0,32,0] GTRU SCH$C_CUR
: 986      1045      3      THEN
: 987      1046      3      STATE_PTR = STATES [0]      ! Illegal
: 988      1047      3      ELSE
: 989      1048      3      STATE_PTR = STATES [2*.RET_NUMBER [0,0,32,0]];
: 990      1049      3      RET_LENGTH [0] = .STATE_PTR [0];
: 991      1050      3      CH$MOVE (.RET_LENGTH [0], STATE_PTR [1], RET_STRING [0]);
: 992      1051      3      END;
: 993      1052      3
: 994      1053      3      [LIB$K_FMT_MODE]:
: 995      1054      3      BEGIN
: 996      1055      3      LOCAL
: 997      1056      3      MODE_PTR: REF VECTOR [, BYTE];
: 998      1057      3      MODE_PTR = MODES [0];
: 999      1058      3      INCR I FROM 1 TO .RET_NUMBER [0,0,32,0] DO
1000      1059      4      BEGIN
1001      1060      4      IF .MODE_PTR [0] EQLU 0
1002      1061      4      THEN
1003      1062      5      BEGIN
1004      1063      5      MODE_PTR = MODES [0];      ! Invalid, use OTHER
1005      1064      5      EXIT[OOP];
1006      1065      4      END;
1007      1066      4      MODE_PTR = .MODE_PTR + .MODE_PTR [0] + 1; ! Skip this string
1008      1067      4      END;
1009      1068      3      RET_LENGTH [0] = .MODE_PTR [0];
1010      1069      3      CH$MOVE (.RET_LENGTH [0], MODE_PTR [1], RET_STRING [0]);
1011      1070      3      END;
1012      1071      3
1013      1072      3      [LIB$K_FMT_PSTRING,LIB$K_FMT_ASCIC]:      ! Strip trailing blanks
1014      1073      3      BEGIN
1015      1074      3      DECRU I FROM .RET_LENGTH [0] TO 1 DO
1016      1075      3      IF .RET_STRING [I] EQL %C' '
1017      1076      3      THEN
1018      1077      3      RET_LENGTH [0] = .RET_LENGTH [0] - 1;
1019      1078      3      END;
1020      1079      3
1021      1080      3      Note: $GETJPI does not return counted strings. It returns
1022      1081      3      these strings as zero-padded strings.
1023      1082      3
1024      1083      3      [LIB$K_FMT_ASCIC]:
1025      1084      3      BEGIN
1026      1085      3      RET_LENGTH [0] = .RET_STRING [0];
1027      1086      3      CH$MOVE (.RET_LENGTH [0], RET_STRING [1], RET_STRING [0]);
1028      1087      3      END;
1029      1088      3
```


: 1030
: 1031
: 1032
: 1033
: 1034
: 1035
: 1036
: 10371089 2
1090 2
1091 2
1092 2
1093 2
1094 2
1095 2
1096 1[INRANGE, OUTRANGE]:
;
TES;
RETURN;
END;

! End of routine LIB\$\$FORMAT_RESULT;

4E	57	4F	4E	4B	4E	55	07	001B0	P.AAA:	.ASCII	<7>\UNKNOWN\				
		31	56	31	31	46	05	001B8		.ASCII	<5>\F11V1\				
						00	00	001BE		.BYTE	0 0				
		32	56	31	31	46	05	001C0		.ASCII	<5>\F11V2\				
						00	00	001C6		.BYTE	0 0				
				41	54	4D	03	001C8		.ASCII	<3>\MTA\				
				00	00	00	00	001CC		.BYTE	0 0 0 0				
				54	45	4E	03	001D0		.ASCII	<3>\NET\				
				00	00	00	00	001D4		.BYTE	0 0 0 0				
				4D	45	52	03	001D8		.ASCII	<3>\REM\				
				00	00	00	00	001DC		.BYTE	0 0 0 0				
				4C	4E	4A	03	001E0		.ASCII	<3>\JNL\				
				00	00	00	00	001E4		.BYTE	0 0 0 0				
4E	57	4F	4E	4B	4E	55	07	001E8	P.AAB:	.ASCII	<7>\UNKNOWN\				
		47	50	4C	4F	43	05	001F0		.ASCII	<5>\COLPG\				
						00	00	001F6		.BYTE	0 0				
		54	49	41	57	4D	05	001F8		.ASCII	<5>\MWAIT\				
						00	00	001FE		.BYTE	0 0				
				46	45	43	03	00200		.ASCII	<3>\CEF\				
				00	00	00	00	00204		.BYTE	0 0 0 0				
				57	46	50	03	00208		.ASCII	<3>\PFW\				
				00	00	00	00	0020C		.BYTE	0 0 0 0				
				46	45	4C	03	00210		.ASCII	<3>\LEF\				
				00	00	00	00	00214		.BYTE	0 0 0 0				
		4F	46	45	4C	04	00218		.ASCII	<4>\LEFO\					
				00	00	00	0021D		.BYTE	0 0 0					
				42	49	48	03	00220		.ASCII	<3>\HIB\				
				00	00	00	00	00224		.BYTE	0 0 0 0				
		4F	42	49	48	04	00228		.ASCII	<4>\HIBO\					
				00	00	00	0022D		.BYTE	0 0 0					
		50	53	55	53	04	00230		.ASCII	<4>\SUSP\					
				00	00	00	00235		.BYTE	0 0 0					
		4F	50	53	55	53	05	00238		.ASCII	<5>\SUSPO\				
						00	00	0023E		.BYTE	0 0				
				47	50	46	03	00240		.ASCII	<3>\FPG\				
				00	00	00	00	00244		.BYTE	0 0 0 0				
				4D	4F	43	03	00248		.ASCII	<3>\COM\				
				00	00	00	00	0024C		.BYTE	0 0 0 0				
		4F	4D	4F	43	04	00250		.ASCII	<4>\COMO\					
						00	00	00255		.BYTE	0 0 0				
				52	55	43	03	00258		.ASCII	<3>\CUR\				
				00	00	00	00	0025C		.BYTE	0 0 0 0				
		52	45	48	54	4F	05	00260	P.AAC:	.ASCII	<5>\OTHER\				
4B	52	4F	57	54	45	4E	07	00266		.ASCII	<7>\NETWORK\				
		48	43	54	41	42	05	0026E		.ASCII	<5>\BATCH\				
45	56	49	54	43	41	52	45	54	4E	49	0B	00274		.ASCII	<11>\INTERACTIVE\

```

ACP TYPES= P.AAA
STATES= P.AAB
MODES= P.AAC
LIB$$SAT_PRV_NAMES== P.AAD

```


[illegible]

		E0	AD	08	AC	D0	000B6	9\$:	MOVL	RET_NUMBER, PRMLST	9926	
					0097	31	000BB	10\$:	BRW	23\$	9930	
		53			56	D0	000BE		MOVL	R6, STRING_PTR	9940	
		57		00D1	CA	9E	000C1		MOVAB	LIB\$\$AT_PRV_NAMES, PRV_NAME	9941	
					59	D4	000C6		CLRL	PRV	9947	
		58			67	9A	000C8	11\$:	MOVZBL	(PRV_NAME), R8	9944	
					19	13	000CB		BEQL	13\$		
08	08	BC			59	E1	000CD		BBC	PRV, @RET_NUMBER, 12\$	9947	
63	01	A7			58	28	000D2		MOV C3	R8, 1(PRV_NAME), (STRING_PTR)	9951	
		83			2C	90	000D7		MOV B	#4, (STRING_PTR)+	9952	
		57		01	A8	9E	000DA	12\$:	MOVAB	1(R8)[PRV_NAME], PRV_NAME	9954	
					59	D6	000DF		INCL	PRV	9942	
		3F			59	D1	000E1		CMPL	PRV, #63		
					E2	1B	000E4		BLEQU	11\$		
		56			53	D1	000E6	13\$:	CMPL	STRING_PTR, R6	9956	
					02	13	000E9		BEQL	14\$		
					53	D7	000EB		DECL	STRING_PTR	9958	
0C	BC		53		56	A3	000ED	14\$:	SUBW3	R6, STRING_PTR, @RET_LENGTH	9959	
						04	000F2		RET		9866	
		F8	AD		03	B0	000F3	15\$:	MOVW	#3, CTRSTR_DESCR	9964	
		FC	AD	01FA	CA	9E	000F7		MOVAB	P.AAJ, CTRSTR_DESCR+4	9965	
					FF	7D	31	000FD	BRW	6\$	9966	
		0E		10	AC	D1	00100	16\$:	CMPL	RET_TYPE, #14	9985	
					07	12	00104		BNEQ	17\$		
		55		01FD	CA	9E	00106		MOVAB	P.AAK, PROT_CHARS	9987	
					05	11	0010B		BRB	18\$		
		55		0201	CA	9E	0010D	17\$:	MOVAB	P.AAL, PROT_CHARS	9989	
		54		C8	AD	9E	00112	18\$:	MOVAB	PSTRING, PSTRING_PTR	9991	
					50	D4	00116		CLRL	I	9999	
		E0	AD	40	54	D0	00118	19\$:	MOVL	PSTRING_PTR, PRMLST[1]	9996	
					54	D0	0011D		MOVL	PSTRING_PTR, THIS_STRING	9997	
		53			84	94	00120		CLRB	(PSTRING_PTR)+	9998	
52	08	BC			02	78	00122		ASHL	#2, I, R3	9999	
		57			53	EF	00126		EXTZV	R3, #4, @RET_NUMBER, R2		
					0F	8D	0012C		XORB3	#15, R2, PROT_FIELD		
					15	13	00130		BEQL	22\$	1000	
					3D	90	00132		MOVB	#61, (PSTRING_PTR)+	1003	
					61	96	00135		INCB	(THIS_STRING)	1004	
					52	D4	00137		CLRL	J	1005	
		06			52	E1	00139	20\$:	BBC	J, PROT_FIELD, 21\$	1007	
					62	45	90	0013D	MOVB	(J)[PROT_CHARS], (PSTRING_PTR)+	1010	
					61	96	00141		INCB	(THIS_STRING)	1011	
					03	F3	00143	21\$:	AOBLEQ	#3, J, 20\$	1005	
		F2			03	F3	00147	22\$:	AOBLEQ	#3, I, 19\$	9992	
		CD			24	B0	0014B		MOVW	#36, CTRSTR_DESCR	1016	
					CA	9E	0014F		MOVAB	P.AAM, CTRSTR_DESCR+4	1019	
		F8	AD	0205	AD	9F	00155	23\$:	PUSHAB	PRMLST	1023	
					FO	AD	9F	00158	PUSHAB	OUTSTR_DESCR		
					OC	AC	DD	0015B	PUSHL	RET_LENGTH		
					F8	AD	9F	0015E	PUSHAB	CTRSTR_DESCR		
		00000000G	00		04	FB	00161		CALLS	#4, SYS\$FAOL		
						04	00168		RET		9866	
					08	BC	D1	00169	24\$:	CMPL	@RET_NUMBER, #6	1031
					05	1B	0016D		BLEQU	25\$		
					51	6A	9E	0016F	MOVAB	ACP_TYPES, ACPTYP_PTR	1033	
					21	11	00172		BRB	28\$		
50	08	BC			01	78	00174	25\$:	ASHL	#1, @RET_NUMBER, R0	1035	

		51		6A40	DE	00179	MOVAL	ACP_TYPES[R0], ACPTYP_PTR		
				16	11	0017D	BRB	28\$	1036	
		0E	08	BC	D1	0017F	26\$:	CMP	@RET_NUMBER, #14	1044
				06	1B	00183		BLEQU	27\$	
		51	38	AA	9E	00185	MOVAB	STATES, STATE_PTR	1046	
				0A	11	00189	BRB	28\$		
50	08	BC		01	78	0018B	27\$:	ASHL	#1, @RET_NUMBER, R0	1048
		51	38	AA40	DE	00190	MOVAL	STATES[R0], STATE_PTR		
	0C	BC		61	9B	00195	28\$:	MOVZBW	(STATE_PTR), @RET_LENGTH	1049
66	01	A1	0C	BC	28	00199	MOVZC3	@RET_LENGTH, 1(STATE_PTR), (R6)	1050	
				04	0019F		RET		0866	
		50	00B0	CA	9E	001A0	29\$:	MOVAB	MODES, MODE_PTR	1057
		52		01	D0	001A5		MOVL	#1, I	1058
				15	11	001A8		BRB	32\$	
				60	95	001AA	30\$:	TSTB	(MODE_PTR)	1060
				07	12	001AC		BNEQ	31\$	
		50	00B0	CA	9E	001AE		MOVAB	MODES, MODE_PTR	1063
				10	11	001B3		BRB	33\$	1062
		51		60	9A	001B5	31\$:	MOVZBL	(MODE_PTR), R1	1066
		50	01	A140	9E	001B8		MOVAB	1(R1)[MODE_PTR], MODE_PTR	
				52	D6	001BD		INCL	I	1058
	08	BC		52	D1	001BF	32\$:	CMP	I, @RET_NUMBER	
				E5	1B	001C3		BLEQU	30\$	
	0C	BC		60	9B	001C5	33\$:	MOVZBW	(MODE_PTR), @RET_LENGTH	1068
66	01	A0	0C	BC	28	001C9	MOVZC3	@RET_LENGTH, 1(MODE_PTR), (R6)	1069	
				04	001CF		RET		0866	
		50	0C	BC	3C	001D0	34\$:	MOVZWL	@RET_LENGTH, I	1074
				0B	11	001D4		BRB	37\$	
		20		6046	91	001D6	35\$:	CMPB	(I)[R6], #32	1075
				03	12	001DA		BNEQ	36\$	
			0C	BC	B7	001DC		DECW	@RET_LENGTH	1077
				50	D7	001DF	36\$:	DECL	I	1075
				F3	12	001E1	37\$:	BNEQ	35\$	
				04	001E3		38\$:	RET		1096

; Routine Size: 484 bytes, Routine Base: _LIB\$CODE + 03D9

LIB\$\$LEXICAL Internal routines for lexical functions
1-009 LIB\$\$FORMAT_RESULT - Format the result

B 6
16-Sep-1984 01:04:32 VAX-11 Bliss-32 V4.0-742
14-Sep-1984 12:39:06 [LIBRTL.SRC]LIBLEXICA.B32;1

Page 34
(13)

: 1039 1097 1 END
: 1040 1098 1
: 1041 1099 0 ELUDOM

! End of module LIB\$\$LEXICAL

PSECT SUMMARY

Name	Bytes	Attributes
_LIB\$CODE	1469	NOVEC,NOWRT, RD , EXE, SHR, LCL, REL, CON, PIC,ALIGN(2)

Library Statistics

File	----- Total	Symbols Loaded	----- Percent	Pages Mapped	Processing Time
_\$255\$DUA28:[SYSLIB]LIB.L32;1	18619	78	0	1000	00:01.4
_\$255\$DUA28:[LIBRTL.OBJ]RTLLIB.L32;1	36	17	47	8	00:00.1

COMMAND QUALIFIERS

BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/NOTRACE/LIS=LI\$:LIBLEXICA/OBJ=OBJ\$:LIBLEXICA MSRC\$:LIBLEXICA/UPDATE=(ENH\$:LIBLEXICA)

: Size: 916 code + 553 data bytes
: Run Time: 00:20.2
: Elapsed Time: 01:30.6
: Lines/CPU Min: 3262
: Lexemes/CPU-Min: 42347
: Memory Used: 229 pages
: Compilation Complete

0208 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

